Jurnal Fisika Indonesia, Nomor 6, Vol.II. Edisi Agustus 1998, ISSN : 1410-2994, hal 1 - 13

PELAYAN DIREKTORI TERDISTRIBUSI

Abdul Ro'uf[1]
Lab. Elektronika, FMIPA, UGM

INTISARI

Telah dibuat satu model Pelayan Direktori Terdistribusi yang berfungsi untuk melayani permintaan arsip didalam sistem terdistribusi, dengan menggunakan memori utama sebagai media untuk menyimpan arsip yang sedang dioperasikan. Ruang memori yang dipakai, lokasinya tersebar di dalam sistem. Model yang dibuat telah diimplementasikan menggunakan mesin Hawlett-Packard dengan dua buah stasiun kerja, yang fasilitas memorinya masing-masing 4 mega-byte dan 8 mega-byte. Dari hasil uji coba, diperoleh bahwa kecepatan akses arsip menjadi lebih cepat dari pada akses langsung ke disk.

Kata-kata kunci : Pelayan, Direktrori, Sistem terdistribusi, Arsip, Cache

DISTRIBUTED DIRECTORY SERVER

Abdul Ro'uf[2]

ABSTRACT

It has been designed and implemented the prototype of the distributed directory server to provide the service of a file access in distributed systems environment. It uses the part of main memory to cache a file. The location of the cache is distributed. The prototype has been implemented and tested on a Hawlett-Packard machine with two workstations. The main memori capacity of each workstation is 8 megabyte and 4 megabyte. The prototype was tested and it shows that the file access will be faster than direct access to the disk.

Keywords : Server, Directory, Distributed systems, File, Cache

I. PENDAHULUAN

Perkembangan mutakhir pada sistem jaringan komputer, khususnya jaringan komputer lokal sangat menggembirakan: yaitu semakin besarnya kapasitas memori yang ada pada masing-masing stasiun kerja (*workstation*). Kapasitas memori yang cukup besar dapat mendukung kecepatan proses-proses komputasi, yang pada akhirnya akan meningkatkan unjuk-kerja sistem secara keseluruhan.

Meskipun ruang memori yang tersedia cukup besar dan juga kecepatan prosesor semakin meningkat, tetapi proses-proses yang menyangkut akses arsip (*file access*) ke disk masih tetap lambat. Hal ini karena masih lambatnya waktu akses disk. Proses dapat dipercepat jika semua data yang diperlukan berada di memori.

Karena itu perlu dipikirkan cara pemanfaatan memori yang tak terpakai yang letaknya tersebar di dalam sistem jaringan tersebut, untuk mendukung proses-proses yang memerlukan akses arsip. Dengan demikian proses akses arsip tersebut dapat dilayani secara lebih cepat.

Untuk dapat memanfaatkan memori yang sedang tak terpakai pada stasiun kerja-stasiun kerja yang ada, diperlukan suatu proses yang dapat berfungsi untuk "mengambil" sebagian ruang memori dari stasiun kerja yang ada kemudian mengelolanya secara baik. Memori yang diperoleh dari semua stasiun kerja yang ada, kemudian dikelola menjadi satu kesatuan untuk mendukung operasi akses arsip, sehingga operasi akses arsip dapat dilayani lebih cepat.

Untuk mengendalikan ruang memori yang diperoleh, dapat dilakukan dengan membagi ruang memori tersebut ke dalam unit-alokasi. Kemudian dibuat tabel-tabel yang mencatat status setiap unit alokasi beserta isinya. Semua tabel-tabel yang ada tersebut akan dikelola oleh suatu proses yang bertugas sebagai koordinator. Maka terbentuklah tabel-tabel yang tersebar dan terkoordinir, tabel-tabel ini dinamakan direktori terdistribusi. Proses yang bertugas melayani permintaan akses arsip dengan memanfaatkan memori sebagai tempat untuk menyimpan sementara arsip yang sedang diakses, serta

melakukan pengelolaan ruang memori menggunakan direktori terdistribusi dinamakan Pelayan Direktori Terdistribusi.

Pelayan direktori terdistribusi ini, akan menjalankan fungsinya seperti fungsi dari sistem arsip terdistribusi (*distributed file system*). Fungsi dari sistem arsip secara keseluruhan sangatlah banyak. Pada pelayan yang dibuat, hanya akan melakukan sebagian dari fungsi sistem arsip, yaitu melayani permintaan operasi arsip yang dasar. Operasi arsip dasar yang dapat dilayani yaitu: operasi buka arsip (*open file*), operasi baca arsip (*read file*), operasi tulis arsip (*write file*), dan operasi tutup arsip (*close file*).

II. DASAR TEORI (TINJAUAN PUSTAKA)

Pelayan direktori terdistribusi yang dibuat ditujukan untuk memberikan fasilitas pelayanan arsip agar arsip-arsip yang dapat digunakan bersama (*file sharing*), dapat diakses secara cepat dengan cara memanfaatkan ruang memori sebagai ruang penyembunyian arsip (*file chaching*) yang bersifat global. Secara umum, tujuan ini sama seperti tujuan dari sistem arsip terdistribusi. Sampai sekarang sudah banyak model sistem arsip terdistribusi yang telah dikembangkan. Beberapa pokok persoalan yang perlu dikaji dalam merancang suatu sistem file terdistribusi antara lain adalah: penamaan dan lokasi, replikasi data, konsistensi cache, dan peremajaan data di pelayan (*server*).

Semua sistem arsip terdistribusi yang ada, memberikan cara penamaan yang bertingkat, baik untuk arsip lokal maupun arsip jauh. Dalam banyak hal struktur penamaan serupa dengan penamaan pada sistem UNIX. CFS (Gifford, 1988) memberi tambahan nomor versi kepada nama. Andrew (Satyanayaranan, 1985) memberikan struktur nama yang *Unix-like* pada antarmuka klien-pelayan (*client-server*) dan membiarkan struktur nama terbuka pada stasiun kerja. Pada NFS (Sanberg, 1985), klien dapat menempelkan (*mount*) setiap sub-directory dari sistem arsip jauh (*remote file system*) pada sistem arsip lokal, ini berarti klien dapat memiliki nama sistem arsip yang berbeda karena ditempelkan dalam

tempat yang berbeda. Amoeba (Mullender, 1985) mengimplementasikan sistem arsip sebagai *tree of pages*, dengan *subtree* berupa arsip-arsip.

Tujuan dari cara-cara penamaan adalah untuk memberikan transparansi lokasi (*location-transparancy*) kepada pemakai. Pada metode Andrew tidak dapat diketahui lokasi suatu arsip dengan melihat namanya. CFS menambahkan lokasi penyimpan (*storage-site*) pada nama suatu arsip; sehingga diketahui di pelayan mana arsip tersebut disimpan (Gifford, 1988). NFS menggunakan cara lain untuk memberikan akses secara transparan (*transparant-access*) yaitu dengan cara melokalkan arsip jauh (*locating-remote-file*) (Sanberg, 1985).

Replikasi data pada lokasi (*site*) yang berbeda pada sistem terdistribusi memberikan 2 keuntungan. Pertama, meningkatkan ketersediaan arsip dengan membolehkan pemakaian salinan-cadangan pada saat arsip utama tidak tersedia. Kedua, memberikan unjuk kerja yang lebih baik dengan dimungkinkannya data diakses dengan waktu akses yang lebih kecil.

Andrew memakai cara penyembunyian (*caching*) sebagai bentuk dari replikasi data (*replication*). Andrew menyembunyikan arsip secara utuh (*whole-file*) dalam disk lokal (Mullender, 1985). CFS menggunakan sebagian dari disk lokal yang tidak ditempati oleh arsip lokal sebagai ruang penyembunyian untuk arsip jauh. Juga memakai cara penyembunyian arsip utuh. CFS juga memakai penyangga cache pada sisi klien, sedangkan sistem Amoeba dan Sprite menggunakan cache baik pada pelayan maupun pada klien (Gifford, 1988).

Adalah sangat mungkin data dalam cache tidak sesuai lagi dengan data pada arsip jauh, hal ini terjadi karena arsip jauh telah berubah. Maka harus dilakukan cara agar cache selalu berisi data yang terbaru atau selalu konsisten dengan data jauh. Barbagai cara untuk menjaga cache agar selalu konsisten telah dilakukan dalam berbagai sistem arsip. ITC (Satyanayaranan, 1985) menyembunyikan arsip utuh yang baru saja diakses (*recently accessed*) dan algoritma untuk konsistensi cache dipanggil pada saat operasi arsip pada cache. Ketika arsip dibuka, ITC memberi jaminan bahwa yang dibuka adalah arsip versi terakhir yang disimpan dalam pelayan jauh (*remote server*). Jadi ITC menjamin konsistensi cache pada tingkat arsip saja (Howard, 1988).

NFS memakai memori utama untuk menyembunyikan blok-blok dari arsip jauh yang baru saja diakses. Blok arsip jauh yang baru saja ditulis akan tetap dijaga dalam memori utama paling lama 30 detik sebelum ditulis kembali ke pelayan, sedang blok arsip jauh yang telah dibaca akan ada di memori utama paling lama 3 detik jika pelayan berisi informasi yang baru. Dengan demikian, jeda (*delay*) maksimum antara waktu informasi ditulis ke stasiun kerja dan menjadi terlihat ke stasiun kerja lain kira-kira adalah 33 detik (30 detik data sampai di pelayan dan 3 detik untuk stasiun kerja lain membaca dari pelayan) (Sanberg, 1985).

CFS memakai cara lain. Dalam CFS, arsip jauh adalah *immutable*. Jadi sekali arsip disembunyikan maka ia tidak pernah berubah, dengan demikian tidak ada masalah konsistensi cache. CFS adalah sistem yang menggunakan cara akses tingkat arsip yang berhubungan secara langsung dengan arsip bersama yang immutable. Dalam CFS, peremajaan pada arsip bersama dicapai dengan menciptakan versi baru dari arsip (Gifford, 1988). Persoalan yang muncul adalah pengelolaan dari versi-versi arsip, dan hal itu lebih mudah ditangani dengan perangkat lunak aplikasi.

Pada sistem Amoeba, baik pelayan maupun klien dapat memiliki cache. Cache berisi versi-versi dari sejumlah arsip, yang berisi sekumpulan halaman (*page*). Ketika klien meminta pelayan untuk mencipta versi baru dari arsip, klien atau pelayan atau keduanya memeriksa cache-nya untuk melihat apakah ada halaman dari versi sebelumnya dari suatu arsip yang masih dipakai. Keutuhan cache hanya perlu diperiksa pada saat mulai transaksi. Untuk mencegah terjadinya konflik dalam peremajaan yang bersamaan digunakan mekanisme penguncian (*locking*) pada tingkat blok (Mullender, 1985).

Pada Sprite, dijamin bahwa setiap pembacaan data pasti diberikan data yang terbaru. Tetapi mekanisme konsistensi cache tidak menjamin bahwa operasi pembacaan dan penulisan yang bersamaan oleh aplikasi selalu terjadi dengan urutan yang benar. Jika urutan tidak benar, aplikasi harus mensinkronkan operasi tersebut dengan memakai cara penguncian arsip atau

memakai mekanisme komunikasi lainnya yang tersedia (Nelson, 1988). Pelayan menjadi pusat kendali untuk menjaga konsistensi cache.

Cara yang dipakai untuk menulis balik data yang ada di cache ke pelayan mempunyai akibat yang kritis terhadap unjuk kerja sistem dan kehandalannya. Cara-cara yang dipakai perlu dipertimbangkan masak-masak terutama dalam menghadapi kegagalan sistem. Pada dasarnya ada dua cara yang dapat dipakai yaitu *write-through* dan *write-back*.

Write-through berarti penulisan ke pelayan dilakukan segera setelah dilakukan penulisan ke cache. Keuntungan cara ini adalah kehandalan sistem, hanya sedikit informasi yang hilang jika terjadi crash. Tetapi akses penulisan menjadi lambat karena penulisan berikutnya hanya dapat dilakukan jika penulisan sebelumnya telah selesai dilakukan, termasuk penulisan ke pelayan. Hal ini berakibat jeleknya unjuk kerja penulisan. Penelitian (Nelson, 1988) menunjukkan bahwa sepertiga dari akses-arsip adalah penulisan, karena itu peremajaan pelayan yang menggunakan cara ini, akan mengurangi traffic ke pelayan kurang lebih dari se-pertiganya.

Cara write-back berarti penulisan ke pelayan dilakukan setelah arsip selesai digunakan (pada saat tutup arsip), karena itu juga disebut *write-on-close*. Cara ini digunakan pada Andrew dan NFS. Jika cache yang digunakan adalah memori utama, jelas bahwa cara ini sangat rawan terhadap kegagalan (kegagalan klien). Di samping itu penelitian terhadap sistem BSD (Nelson, 1988) menunjukkan bahwa 75 persen dari arsip dibuka kurang dari 0,5 detik dan 90 persen dibuka kurang dari 10 detik. Hal ini menunjukkan bahwa cara write-back tidak akan mengurangi *traffic* ke pelayan secara berarti.

Cara lain yang merupakan kombinasi dari kedua cara di atas adalah *delayed-write-back*; yaitu: mula-mula blok-blok ditulis ke cache kemudian ditulis ke pelayan beberapa saat kemudian. Cara ini punya kelebihan dibanding write-through. Yang pertama, karena penulisan hanya dilakukan ke cache maka proses penulisan lebih cepat. Kedua, data dapat dihapus/diganti lagi sebelum ditulis ke pelayan. Penelitian pada BSD menunjukkan bahwa 20 sampai 30 persen data-baru dihapus lagi dalam waktu 30 detik dan 50 persen dihapus

dalam 5 menit. Karena itu jika penulisan ke pelayan ditunda beberapa menit maka akan dapat mengurangi secara berarti traffic ke pelayan. Akan tetapi masalah yang sama pada write-back yaitu kemungkinan hilangnya data karena kegagalan, masih tetap ada. Masalah ini dapat diusahakan penyelesaiannya dengan pemilihan waktu penundaan yang tepat. Misalnya, seperti yang dipakai pada Sprite, semua blok yang tidak diubah dalam waktu 30 detik terakhir, akan ditulis ke pelayan (Nelson, 1988), sehingga akan dapat dihindari kehilangan data jika terjadi kegagalan.

III. PERANCANGAN

Secara fungsional, model yang dibuat terdiri dari 2 lapisan. Lapisan pertama adalah pelayan pusat yang bertugas untuk melayani akses arsip ke disk. Sedangkan lapisan kedua adalah pelayan lokal yang bertugas untuk melayani permintaan arsip oleh pemakai. Pelayan pusat berfungsi sebagai pusat pengendali semua kegiatan yang menyangkut pemakaian ruang cache berkenaan dengan permintaan arsip oleh peminta arsip. Pelayan lokal mengelola ruang cache yang ada di lokal masing-masing. Jika ada perubahan pemakaian ruang cache lokal maka pelayan lokal melaporkan perubahan tersebut ke pelayan pusat. Dengan demikian pelayan pusat setiap saat mengetahui berapa besar ruang cache yang terpakai dan berapa yang masih tersedia dan di mana saja.

Agar pelayan dapat menjalankan fungsinya dengan baik diperlukan tabel-tabel catatan, baik mengenai keadaan cache maupun mengenai arsip-arsip yang sedang dipakai. Struktur dari direktori mengikuti struktur dari pelayan, yaitu mempunyai struktur bertingkat (hirarki) yang terdiri dari tabel pusat dan tabel-tabel lokal. Tabel pusat dikelola oleh pelayan-pusat, dan tabel-lokal masing-masing dikelola oleh pelayan lokal.

Pada tabel pusat informasi mengenai cache akan berisi besarnya cache pada masing-masing stasiun kerja, berapa yang sudah terpakai dan berapa sisanya. Sedangkan informasi mengenai arsip akan berisi nama arsip, lokasi cache dan atribut-atribut lainnya. Lokasi cache dipakai oleh pelayan pusat untuk

m... ...da pelayan lokal yang meminta arsip yang sudah dipakai di ... Dengan begitu pelayan lokal tersebut dapat mengakses pelayan ... di mana arsip tersebut berada.

Pada masing-masing tabel lokal, informasi mengenai cache berisi: banyaknya blok cache yang tersedia dan alamatnya, status dari setiap blok, jumlah blok yang terpakai dan yang bebas. Sedangkan informasi mengenai arsip yang sedang dipakai akan berisi: nama-arsip dan lokasinya serta jumlah pemakainya.

Pelayan direktori terdistribusi melayani permintaan arsip yang terdiri dari: buka arsip, baca arsip, tulis arsip, tutup arsip. Semua jenis permintaan arsip di atas akan diterima oleh pelayan lokal, oleh pelayan lokal permintaan tersebut diperiksa apakah permintaan tersebut perlu diteruskan ke pelayan pusat ataukah tidak, atau mungkin juga pelayan lokal tersebut perlu berkomunikasi dengan pelayan lokal lainnya. Semua itu tergantung pada keadaan arsip yang diminta.

Ada 3 kemungkinan keadaan arsip yang diminta tersebut, yaitu: pertama, arsip yang diminta belum ada yang memakai, kedua, arsip yang diminta sudah dipakai dan berada di cache lokal, ketiga, arsip yang diminta sudah dipakai tetapi tidak berada di cache lokal. Pelayanan yang dilakukan untuk masing-masing jenis permintaan, memperhatikan ke 3 kemungkinan keadaan arsip tersebut di atas.

Akses serentak dapat terjadi dengan 3 kemungkinan: akses baca serentak, akses baca dan tulis serentak, akses tulis serentak. Model yang dibuat dapat menangani akses baca serentak dan akses baca-tulis serentak, dengan beberapa keterbatasan. Akses baca serentak dapat dilayani dengan tanpa menimbulkan masalah. Akses baca dan tulis serentak dilayani dengan cara proses yang melakukan baca akan memperoleh data terbaru sebelum proses tulis. Akses tulis serentak tidak dapat dilayani secara serentak, tetapi akan dilayani secara berurutan.

Data yang ada di cache akan selalu konsisten jika semua pemakai mengacu kepada data yang sama. Hal ini dapat dicapai dengan mudah jika

hanya ada satu salinan data dalam cache. Semua permintaan data mengacu kepada data yang sudah ada di cache tersebut. Pada model yang dibuat, digunakan cara yang demikian.

Data-data dari suatu arsip akan disalin ke cache pada saat operasi buka arsip. Model yang dianut ialah menyalin semua isi arsip ke cache pada saat operasi buka arsip yang pertama. Dengan cara yang demikian diharapkan operasi-operasi baca-tulis pada arsip tersebut akan sangat cepat, dan hal ini yang menjadi salah satu tujuan dibuatnya pelayan direktori terdistribusi ini. Kelemahan dari cara ini adalah jika ruang-cache yang tersedia tidak cukup untuk menampung semua data dari suatu arsip maka permintaan buka arsip tersebut akan gagal.

Untuk meremajakan data di disk cara yang dipilih dalam model ini adalah cara tulis balik yaitu penulisan ke disk akan dilakukan pada saat operasi tutup-arsip oleh pemakai yang terakhir; dan penulisan balik ke disk akan dilakukan jika hanya terjadi perubahan data pada cache. Keuntungan cara ini operasi tulis-arsip akan cepat karena hanya akan dilakukan penulisan ke cache. Kerugiannya, cara ini sangat rawan terhadap kegagalan sistem.

IV. IMPLEMENTASI DAN PENGUJIAN

Model dari pelayan direktori terdistribusi telah di implementasikan di mesin Hawlett-Packard yang dioperasikan dengan sistem operasi HP-UX (UNIX 4.2BSD versi HP) sedangkan bahasa permrogramannya menggunakan bahasa C. Mesin yang digunakan termasuk dalam kategori sistem terdistribusi yang terdiri dari 2 buah stasiun kerja. Satu stasiun kerja sebagai sistem-konsul dan yang lain sebagai stasiun-kerja tanpa disk. Kedua buah stasiun kerja berhubungan melalui jaringan yang menggunakan antarmuka Ethernet. Kapasitas memori pada sistem-konsul sebesar 8 mega-byte, sedangkan pada stasiun kerja tanpa disk kapasitas memorinya sebesar 4 mega-byte.

Untuk mengimplementasikan model yang dirancang diperlukan fasilitas komunikasi antar proses (*interprocess communication, IPC*). Dalam sistem UNIX tersedia beberapa cara untuk keperluan komunikasi antar proses,

antara lain dengan *semaphore*, *pipe*, *message-passing* dan *shared-memory* (Bach, 1986). Jika proses-proses yang saling komunikasi terdistribusi di beberapa mesin yang saling terhubung melalui sistem jaringan, maka komunikasi antar proses tersebut akan melibatkan pengaturan komunikasi antar mesin dalam jaringan, sehingga memerlukan protokol jaringan (Tanenbaum, 1988).

Pada sistem yang digunakan, tersedia fasilitas komunikasi dengan protokol TCP dan UDP. Pada penerapan model yang dipilih digunakan TCP (*Tansport Control Protocol*) karena dengan TCP dijamin bahwa data yang dikirim pasti sampai di tempat yang dituju secara utuh dan dengan urutan yang benar (Comer, 1988).

Pada model yang dirancang terdapat 3 jenis proses, yaitu: proses-pelayan-pusat, proses-pelayan-lokal dan proses-peminta-arsip. Cara komunikasi yang diterapkan menganut model komunikasi klien-pelayan (*client-server model*). Proses yang minta pelayanan dinamakan klien (*client*) dan proses yang melayani permintaan dinamakan pelayan (*server*). Model komunikasi klien-pelayan ini dapat diterapkan secara umum, baik proses-proses tersebut berada dalam satu mesin atau berada pada mesin-mesin yang terpisah di dalam sistem jaringan (Tanenbaum, 1987)

Proses-pelayan-pusat merupakan pelayan bagi proses-proses pelayan-lokal. Proses pelayan lokal merupakan pelayan bagi proses peminta-arsip dan sebagai proses klien bagi proses pelayan pusat dan proses pelayan lokal lainnya. Proses pelayan dibuat sebagai proses yang jalan terus menerus selama proses tersebut tidak dimatikan (di *kill*) atau selama mesin tidak dimatikan, proses yang demikian biasa disebut *daemon*.

Proses daemon dibuat dengan cara membuat proses-anak yang berputar selamanya (*infinite looping*) menunggu permintaan hubungan. Setiap hubungan yang terjadi dapat dibuat proses-anak baru untuk melayani permintaan layanan tertentu, sehingga permintaan hubungan lainnya dapat dilayani lagi tanpa harus menunggu pelayanan sebelumnya berakhir. Dengan demikian proses pelayan dapat melayani beberapa proses klien bersama-sama.

Cara ini diterapkan untuk proses baca arsip yang bersama-sama (*concurrent read*) dan untuk satu proses tulis arsip dan proses-proses lainnya baca arsip (*concurrent read write*). Sedangkan jika ada permintaan pelayanan operasi tulis oleh beberapa proses klien bersama-sama, maka akan dilayani satu demi satu, satu dilayani sampai selesai baru yang lainnya dilayani kemudian.

Cache pada masing-masing stasiun-kerja disusun dalam bentuk blok. Karena itu pada masing-masing pelayan lokal dibuat struktur data larikan yang setiap barisnya berisi komponen-komponen: nomor blok, alamat blok dan status blok. Sedangkan pada pelayan-pusat, hanya diperlukan jumlah blok cache pada masing-masing stasiun kerja. Karena itu pada pelayan pusat dibuat struktur data larikan yang berisi komponen-komponen: jumlah blok, nama lokasi dan jumlah yang bebas.

Untuk keperluan pencatatan arsip-arsip yang sedang dipakai dilakukan pencatatan baik di pelayan-pusat maupun pelayan-lokal. Pelayan pusat hanya mencatat arsip-arsip yang sedang dipakai dan dimana lokasi cache-nya, karena itu struktur data arsip pada pelayan-pusat dibuat berupa *list*, yang masing-masing *node*nya berisi komponen-komponen: nama arsip, lokasi cachenya, ukuran arsip dan id-arsip. Sedangkan tabel arsip pada pelayan-lokal dibuat lebih rinci. Ada 3 macam informasi mengenai arsip, yaitu identitas arsip, identitas pemakai dan lokasi arsip, karena itu stuktur data untuk arsip di pelayan-lokal terdiri dari 3 buah list yang saling berhubungan, yaitu: list id masing-masing pemakai, list arsip, dan list blok arsip.

Secara umum, pesan yang dikirim oleh proses klien ke proses pelayan dibagi dalam 3 *field* (bagian). Setiap *field* diakhiri dengan karakter NULL (0). Field-1 berisi jenis permintaan, sebanyak 1 byte. Sedangkan field-2 dan field-3 berisi data-data yang diperlukan untuk setiap jenis permintaan.

Dari model yang dibuat, telah dilakukan beberapa jenis percobaan. Tujuan dari percobaan adalah untuk mengetahui apakah semua fasilitas layanan yang disediakan dapat berfungsi sesuai dengan yang diharapkan. Disamping itu juga telah dihitung waktu yang diperlukan dalam melayani permintaan akses arsip. Untuk keperluan itu, telah dibuat beberapa program penguji yang dapat

melakukan pengujian terhadap fungsi akses baca arsip secara serempak dan akses baca-tulis secara serempak. Dari pengujian tersebut diperoleh hasil bahwa fungsi layanan yang diberikan telah dapat berfungsi sesuai dengan yang diharapkan.

Sedangkan untuk mendapatkan waktu akses arsip, dibuat program penguji yang dapat menghitung komponen waktu yang memberikan sumbangan pada proses akses arsip, yaitu waktu *setting*, waktu *connect*, waktu *send* dan *receive*. Adapun data-data waktu (dalam mikrodetik) yang diperoleh ditunjukkan dalam tabel I

Tabel I. Waktu Akses Arsip (μs)

Waktu	Data 1	Data 2	Data 3	Rata-rata
Setting	59390	55545	58036	57657
Connect	24258	23026	24295	23859.67
Send	7497	8292	13968	9919
Receive	22165	16292	13886	17447.67
Jumlah	113310	103155	110185	108883.33

Dari pengujian tersebut diperoleh hasil jumlah rata-rata dari komponen waktu akses arsip adalah sebesar 108,9 millidetik. Untuk akses arsip langsung ke disk diperlukan waktu *seek,* waktu rotasi dan waktu transfer, yang pada mesin yang digunakan besarnya adalah 300 millidetik. Jadi dengan pelayan direktori terdistribusi, pelayanan terhadap akses arsip menjadi lebih cepat.

V. KESIMPULAN

Pelayan direktori terdistribusi yang telah dibuat, dapat melayani permintaan akses arsip. Permintaan secara serempak oleh beberapa peminta dapat dilayani secara serempak pula jika jenis permintaannya adalah baca-serempak sedangkan untuk permintaan yang tulis-serempak maka layanan yang diberikan adalah bergantian. Jika kapasitas memori dari stasiun kerja yang digunakan semakin besar maka ukuran cache juga akan semakin besar. Dengan cache yang semakin besar, maka akses data juga akan semakin cepat. Dengan menggunakan memori sebagai tempat untuk menyimpan arsip yang sedang

dioperasikan, telah dapat meningkatkan kecepatan pelayanan akses arsip dengan faktor kecepatan 3 kali dibadingkan jika akses arsip dilakukan langsung ke disk.

DAFTAR PUSTAKA

Bach Maurice, 1986, *The Design and Implimentation of UNIX Operting System*, Prentice Hall Software Series, Englewood Cliffs, New Jersey.

Comer D., 1988, *Internetworking With TCP/IP: Principles, Protocols, and Architectures,* Prentice-Hall Inc, Englewood Cliffs, New Jersey.

Gifford, David K. et. al., 1988, *The Cedar File System*, Communication of the ACM, Vol 31 No. 3, March.

Howard, John H., 1988, *Scale and Performance in a Distributed File System*, ACM trans.on Comp. Sys, Vol.6, No.1, February.

Mullender S.J. et.al., 1985, *A Distributed File Service Based on Optimistic Concurrency Control*, Proc.of the tenth ACM Symposium on O/S Principle, December.

Nelson M.N., et.al., 1988, *Caching in the Sprite Network File System*, ACM trans. On Comp. Sys.,Vol.6, No.1, February.

Satyanarayanan M. et.al, 1985, *The ITC Distributed File System: Principle and Design*, Proc.of.the 10th ACM Symposium on O/S Principle, Washington.

Sanberg Russel.et.al., 1985, *Design and Implementation of the Sun Network File System*, Proc.of USENIX Summer 1985 conference, Oregon, June.

Tanenbaum Andrew S., 1987, *Operating System: Design and Implementation*, Prentice-Hall Inc., Englewood Cliffs, New Jersey.

Tanenbaum Andrew S., 1988., *Computer Networks*, Prentice-Hall Int. Inc., 2-ed, Englewood Cliffs, New Jersey